30 Years of Newest Vertex Bisection

William F. Mitchell [1]

Applied and Computational Mathematics Division, National Institute of Standards and Technology, Gaithersburg, MD, 20899, USA

Abstract. One aspect of adaptive mesh refinement in the finite element method for solving partial differential equations is the method by which elements are refined. In the early 1980's the dominant method for refining triangles was the red-green algorithm of Bank and Sherman. The red refinements are the desired refinements, but will result in an incompatible mesh when used alone. The green refinements are used to recover compatibility for stability of the finite element discretization, and are removed before the next adaptive step. Prof. Bob Skeel raised the question as to whether it is possible to perform adaptive refinement of triangles without this complicated patching/unpatching process. As a result, a new triangle refinement method, called newest vertex bisection, was devised as an alternative to red-green refinement in the mid 1980's. The new approach is simpler and maintains compatibility of the mesh at all times, avoiding the patching/unpatching of the green refinement. In this historical paper we review the development of the newest vertex bisection method for adaptive refinement, and subsequent extensions of the method.

Keywords: finite elements, adaptive mesh refinement, newest vertex bisection
PACS: 02.60.Lj, 02.70.Dh

INTRODUCTION

The numerical solution of partial differential equations is the most computationally intense part of many scientific and engineering applications. Consequently, for many years much research has been devoted to improving the speed and accuracy of the algorithms used for this purpose. In the early 1980's much attention was placed on adaptive mesh refinement in the finite element method. The promise of adaptive mesh refinement was to reduce the number of vertices in the mesh, and consequently the size of the linear systems to be solved, by using small elements only in the areas where the solution is changing rapidly, and large elements where the solution is fairly constant.

One approach to adaptive refinement of triangles at that time was the so-called red-green refinement of Bank and Sherman [1]. Here a red refinement is a quadrisection of a triangle by connecting the midpoints of the sides, and a green refinement is a bisection by connecting a vertex to the midpoint of the opposite side, as in Figure 1. After determining which elements should be refined, those elements are refined via red refinement, most likely resulting in an incompatible mesh with hanging nodes, i.e. vertices in the middle of a side of a triangle. Green refinements are then performed to remove the hanging nodes, resulting in a compatible mesh, i.e. a mesh in which the intersection of any two triangles is either empty, a common vertex, or a common edge. The partial differential equation is discretized with the finite element method on this mesh, and an approximate solution is computed. Then the green refinements are removed and the whole process is repeated until the solution is sufficiently accurate.

When the author began his PhD research under the direction of Prof. Bob Skeel at the University of Illinois Urbana-Champaign in 1985, Prof. Skeel asked the question of whether it would be possible to avoid the green refinement/unrefinement in an adaptive mesh refinement algorithm. The ultimate result of this question was the newest vertex bisection method [2, 3]. In this paper we review the development of the newest vertex bisection method for adaptive refinement, and subsequent extensions of the method. We begin by describing the newest vertex bisection of triangles in the context of adaptive mesh refinement, then look at several approaches to extending newest vertex bisection to tetrahedra in 3D, and finally consider some parallel implementations of newest vertex bisection.

[1] Dedicated to Prof. Bob Skeel on the occasion of his career on Numerical Analysis and Applied Mathematics.

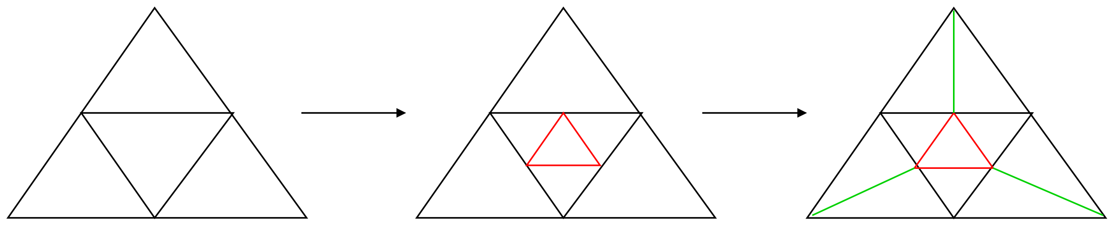

FIGURE 1. Red refinement of a triangle (center) resulting in hanging nodes, and green refinement (right) to enforce compatibility.

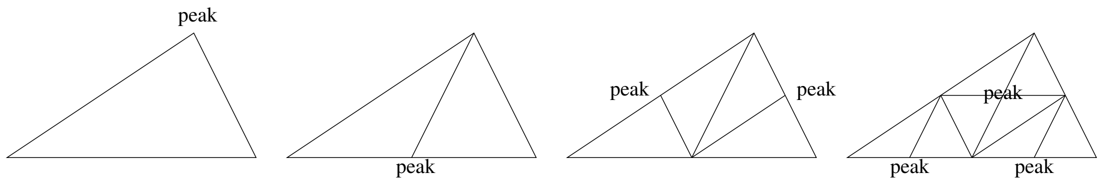

FIGURE 2. Propagation of the peak as the newest vertex.

BISECTION OF TRIANGLES

Newest Vertex Bisection of Triangles

In bisection refinement of triangles, a triangle is refined by connecting a vertex, called the *peak*, to the opposite side, called the *refinement edge*. The two resulting triangles are the *children* of the refined triangle, which is their *parent*, and have *generation* one larger than the parent. Triangles in the initial mesh have generation 1.

A natural question is how to choose the peak. In newest vertex bisection the vertex selected as the peak and the order in which triangles are bisected are very carefully controlled. When a triangle is bisected, a vertex, called the *newest vertex*, is created in the middle of the refinement edge. This vertex is selected as the peak of the two child triangles (Figure 2).

If an arbitrary set of triangles was to be refined, this bisection would result in hanging nodes and further refinements would be needed to make the mesh compatible. But hanging nodes do not arise in two situations (Figure 3):

1. two triangles that share a common refinement edge are refined at the same time, and
2. a triangle whose refinement edge is on the boundary is refined.

A triangle whose refinement edge is either the refinement edge of the triangle that shares it or is part of the boundary is said to be *compatibly divisible*. If two triangles share a common refinement edge, they are called *mates*.

By always refining compatibly divisible triangles, hanging nodes never appear and the mesh is always compatible. But suppose you want to refine a triangle that is not compatibly divisible. It is easily shown [2] that, with newest vertex bisection, if the triangle opposite the peak is refined then the triangle becomes compatibly divisible. This leads to the following recursive algorithm for compatible refinement:

```
procedure refine_triangle(T)
if T is not compatibly divisible then
    refine_triangle(triangle opposite the peak)
end if
bisect T and, if it exists, T's mate
```

The sequence of triangles found by the recursion is called the *compatibility chain*. Figure 4 illustrates an example of refining the compatibility chain. In this figure the peak is always the vertex at the right angle. If every triangle in the initial mesh is compatibly divisible, then this recursion is finite. This is a consequence of the fact that each triangle in the compatibility chain has generation one less than the preceeding triangle in the chain, so the length of the recursion is bounded by the generation of the triangle to be refined.

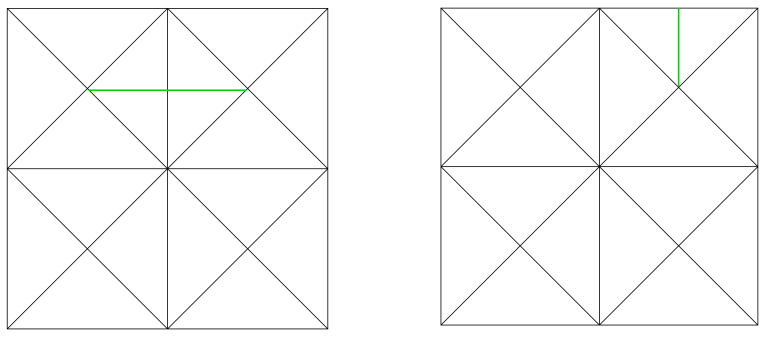

FIGURE 3. Refinement (green line) of a pair of triangles that share a common refinement edge (left) and a triangle whose refinement edge is on the boundary (right).

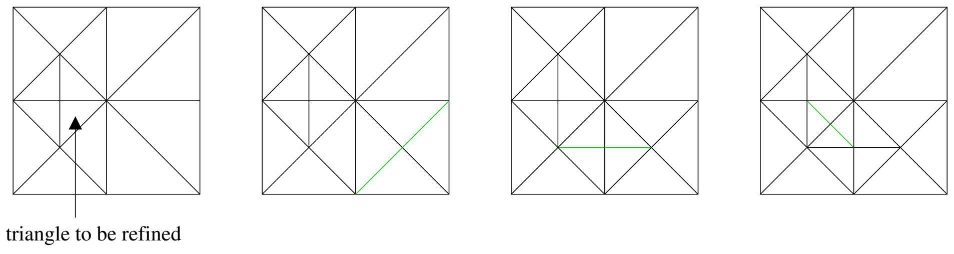

FIGURE 4. Refinement of a triangle that is not compatibly divisible by refining the compatibility chain.

Another important property of a triangle refinement method is that the triangles do not degenerate, i.e. the smallest angle is bounded away from zero. For newest vertex bisection this was proven by Sewell [4]. As shown in Figure 5, there are only four similarity classes among the descendants of a triangle, and thus the angles are guaranteed to be bounded away from 0.

Initial Refinement Edges

The finiteness of the compatibility chain depends on being able to assign the refinement edges of the initial mesh such that every triangle is compatibly divisible. It was proven in [2] that such an assignment exists for any compatible mesh of triangles. This proof relies on a result from graph theory.

The *extended dual graph* of a compatible mesh of triangles is a graph consisting of

- a node for each triangle,

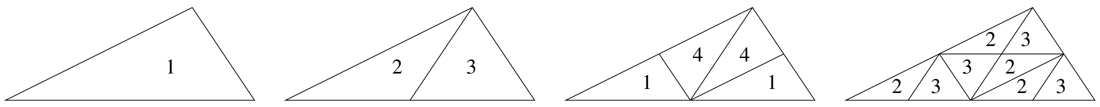

FIGURE 5. The four similarity classes of triangles from newest vertex bisection.

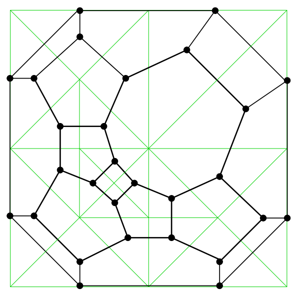

FIGURE 6. An extended dual graph (black) of a compatible mesh of triangles (green).

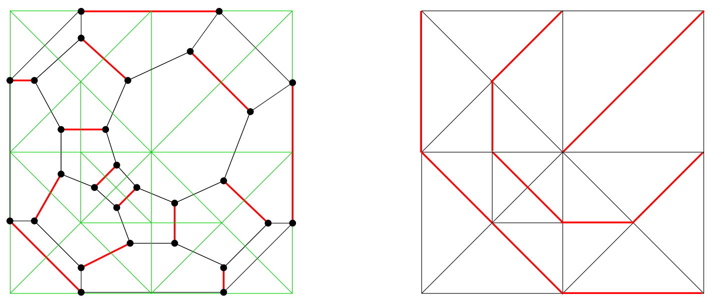

FIGURE 7. A perfect matching (red, left) for an extended dual graph, and the assignment of refinement edges (red, right) in the corresponding mesh.

- a node for each boundary side of a triangle,
- an edge between nodes corresponding to triangles that share a side,
- an edge between nodes corresponding to a boundary side and the triangle that contains it, and
- an edge between nodes corresponding to boundary sides that share a vertex.

Figure 6 illustrates an extended dual graph.

It is easily seen, and can be formally proven [2] that the dual graph is 3-regular (i.e. every node has three edges adjacent to it), and has no cut edges (i.e. has no edge such that the removal of that edge makes the resulting graph disconnected). A well known theorem of Peterson [5] states that every 3-regular graph without cut edges has a perfect matching, i.e. a subset of edges such that every node is on exactly one of those edges (Figure 7, left). This provides a matching, or pairing up, of nodes in the graph. This in turn defines a matching in the mesh (Figure 7, right) where each edge in the perfect matching

1. pairs up two triangles, with their common edge being the refinement edge, or
2. pairs up a triangle and a boundary side of it, which is its refinement edge, or
3. pairs up two boundary edges, which is not used.

This shows that every compatible mesh of triangles has an assignment of refinement edges such that every triangle is compatibly divisible, but is it easy to *find* such an assignment? The perfect matching problem is known to be NP-Complete [6], so the prospects look grim. However, the special case of 3-regular graphs with no cut edges is not NP-Complete. Biedl et al. [7] developed an $O(N)$-time algorithm for finding a perfect matching in such graphs, and used the assignment of initial refinement edges for newest vertex bisection as an example application of their algorithm.

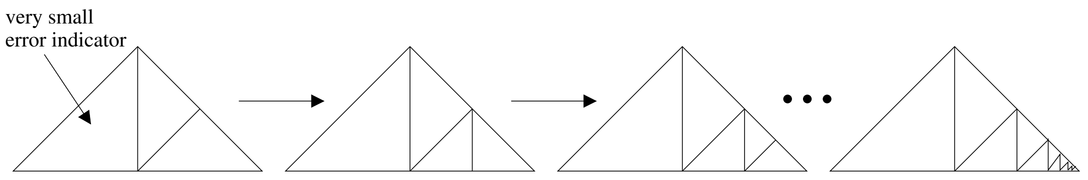

FIGURE 8. Failure of newest vertex bisection without the compatibility chain.

Other Triangle Bisection Methods

Sewell [4] proposed newest vertex bisection refinement of triangles more than 10 years prior to the work described in the previous sections. However, it was not well publicized. It was only written up as his PhD thesis and never published in a refereed journal. He did not address the issue of assigning refinement edges in the initial mesh, and did not have the compatibility chain to allow refinement of triangles that are not compatibly divisible. Instead, he removed those triangles from consideration for refinement. It is possible for that to fail, as in Figure 8. In this example, if the left-most triangle has an error indicator so small that it will never be refined, then it is always the case that the only compatibly divisible triangle is the one that contains the right-most vertex.

Rivara [8] was proposing triangle bisection at about the same time as the work in the previous section. This was not newest vertex bisection, but longest edge bisection. Here the refinement edge is chosen to be a longest edge of the triangle. Refinement is performed in two steps. First the desired triangles are bisected, which likely results in hanging nodes. Then additional bisections are performed to remove the hanging nodes, a process now known as *closure*. This is similar to the first two steps of red-green refinement, except the refinements performed for closure do not need to be removed. Rivara proved that the number of refinements needed for closure is finite. She also proved that longest edge bisection of triangles results in a finite number of similarity classes, so the triangles will not degenerate, but the number of similarity classes depends on the shape of the initial triangles and can be more than four.

BISECTION OF TETRAHEDRA AND n-SIMPLICES

Several researchers have extended the newest vertex bisection of triangles to higher dimensions, i.e. the tetrahedron in $\mathbb{R}^3$ and the n-simplex in $\mathbb{R}^n$. In dimensions greater than 2, selection of the newest vertex does not uniquely define the bisection of the simplex, because there is not a unique edge opposite the newest vertex. Instead, the bisection algorithms focus on designating a refinement edge. Bisection of a tetrahedron means creating a new vertex at the midpoint of the refinement edge, and dividing the tetrahedron by the plane that passes through the new vertex and the two vertices opposite the refinement edge. Methods for which the refinement edge happens to always be one of the edges opposite the newest vertex can be called newest vertex bisection.

In 1984, Rivara [8] considered longest edge bisection of n-simplices, but then presented details only for the triangle. The details for longest edge bisection of tetrahedra were given in 1992 [9]. Like the longest edge bisection of triangles in the previous section, this is a two step process with the second step being closure via additional bisections. Finiteness of closure is proven. The paper contains an empirical study of the minimum angle, but there is no proof that the minimum angle is bounded away from 0.

Bänsch [10] presented the first newest vertex tetrahedron bisection method in 1991. In this method a refinement edge is assigned to each face via triangle newest vertex bisection, and the refinement edge of a tetrahedron is assigned to be an edge that is a refinement edge of two of the faces of the tetrahedron. Bänsch proposes a two step process with closure via additional bisections, and proves finiteness of the closure. The paper does not address any quality metrics of the resulting tetrahedra, such as finiteness of the number of similarity classes.

In 1994, Kossaczky [11] presented a recursive newest vertex bisection algorithm for tetrahedra. The recursion defines the compatibility chain for closure, the same as the newest vertex bisection of triangles in the previous section. This algorithm begins by assuming six tetrahedra are embedded in a parallelepiped, P, which limits the initial mesh to those that satisfy this property. Figure 9 illustrates the sequence of bisections that this algorithm produces. In each image the red tetrahedron is the tetrahedron being bisected, and the blue edges are the edges added by bisection. In the

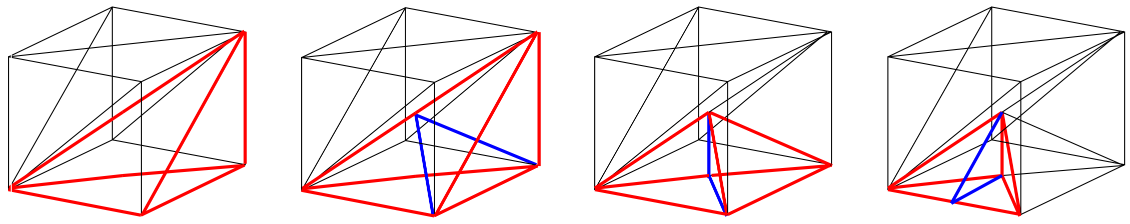

FIGURE 9. Sequence of refinements from Kossaczky's algorithm.

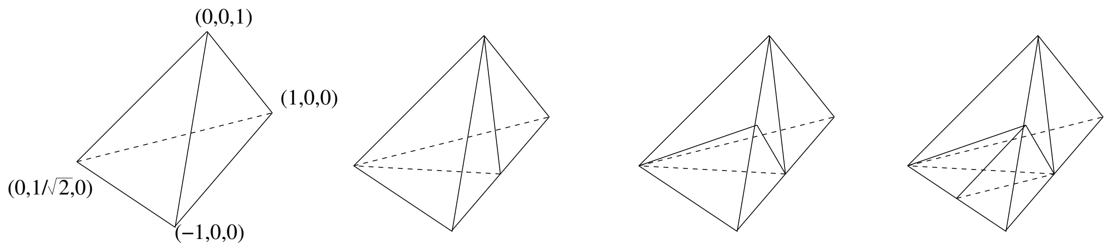

FIGURE 10. The canonical tetrahedron of Liu and Joe, and its refinement sequence.

first refinement, the refinement edge is the diagonal of P, in the second it is the diagonal of a face of P, and in the third it is an edge of P. After three (uniform) refinements, P has been refined to eight parallelepipeds similar to P, each with six embedded tetrahedra, and the sequence of refinements can be repeated. Kossaczky proves there is a finite number of similarity classes, the length of the compatibility chain is finite, and that this method is equivalent to Bänsch's in the sense that it produces the same tetrahedra shapes.

In 1995 Liu and Joe [12] presented an algorithm that, in principle, maps a tetrahedron to a canonical tetrahedron (Figure 10), performs longest edge bisection in the canonical tetrahedron, and maps the result back to the original tetrahedron. This is not the same as longest edge bisection in the original tetrahedron, and is, in fact, a newest vertex bisection. After three refinements in the canonical tetrahedron, all the children are similar to the canonical tetrahedron, so this algorithm also cycles through three types of refinements, as shown in Figure 10. In practice, the mapping is not actually performed, but only indicates the assignment of the refinement edges. They use the two step approach with additional refinements for closure, and prove the finiteness of the closure. In addition to bounding the number of similarity classes by 168, they prove that another tetrahedron quality metric, the mean ratio, is bounded away from zero. They also prove that the generation of adjacent tetrahedra differ by at most two.

Also in 1995, Maubach [13] gave a bisection algorithm for n-simplices that is based on newest vertex bisection of triangles. It requires that the order of the vertices in an n-simplex satisfy a particular property with respect to the vertices of the neighboring simplices. Although not every initial mesh will satisfy this property, he proves that the property can be satisfied if the initial mesh is generated by reflections. To refine an n-simplex $T = x_0, x_1 \dots x_{n-1}, x_n$ with vertices $\{x_i\}_{i=0}^{n}$, let $k = n - \text{generation}(T) \bmod n$, and let $z = (x_0 + x_k)/2$ be the midpoint of the refinement edge (x_0, x_k). The two child simplices are then $x_0, x_1 \dots x_{k-1}, z, x_{k+1} \dots x_n$ and $x_1, x_2 \dots x_k, z, x_{k+1} \dots x_n$. If $n = 2$, this is equivalent to newest vertex bisection of triangles. He uses recursion to define the compatibility chain, and proves it is finite. He also proves there is a finite number of similarity classes.

Two years later, Traxler [14] presented an algorithm for the n-simplex that generates the same simplicies as Maubach, but uses a different ordering of the vertices. In this method $k = \text{generation}(T) \bmod n$ and the refinement edge is (x_0, x_n) with midpoint z. The children are $x_0, z, x_1 \dots x_k \dots x_{n-1}$ and $x_n, z, x_1 \dots x_k, x_{n-1} \dots x_{k+1}$. This method preserves a "structural condition" that says the k^{th} neighbor is either nil (the boundary), one generation lower, or the same generation with a shared refinement edge, a property that newest vertex bisection of triangles also has. He uses recursion to define the compatibility chain, proves it is finite, and proves there is a finite number of similarity classes. There is one restriction on the initial mesh; there must be an even number of simplices around each interior edge.

Finally in 2000, Arnold, Mukherjee and Pouly [15] presented another tetrahedron bisection algorithm. In this

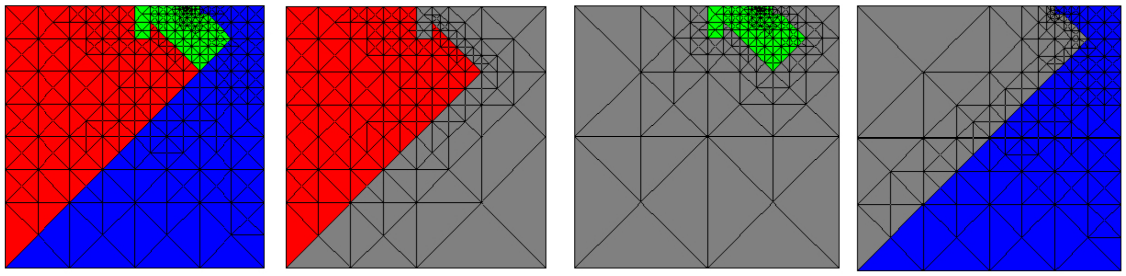

FIGURE 11. An adaptive mesh partitioned for three processors, and the mesh that each processor has when using the full domain partition. Red, green and blue indicate the triangles owned by each processor; gray triangles are ghost triangles.

method each face has a marked edge, and the refinement edge is a marked edge in two faces. There are five types of tetrahedra, depending on the orientation of the marked edges. They give rules for the transition between the types of tetrahedra and the assignment of marked edges when a tetrahedron is bisected. They show the method is essentially equivalent to those of Bänsch and Liu and Joe, and related to Maubach's method. They prove that the number of similarity classes is at most 72. They use the two step approach and prove the closure is finite by giving an upper bound on the generation of tetrahedra generated by closure.

PARALLEL IMPLEMENTATIONS

Independent Sets

Jones and Plassmann [16] introduced a parallel algorithm for adaptive refinement with triangle bisection in 1997. They present the algorithm in the context of longest edge bisection, but note that it is also applicable to newest vertex bisection with slight modifications.

They note that when two triangles that share an edge are refined by different processors, there are two synchronization issues that must be addressed. First, one must not create two different vertices at the same place on the common edge. Second, triangle neighbor information must be maintained. If the two triangles are refined in the same parallel step, a child triangle may think the neighbor on the other processor is the parent, rather than one of the children.

Jones and Plassmann solve the synchronization issues by using independent sets in the dual graph of the mesh. An independent set in a graph is a set of nodes such that no two nodes have an edge between them. The corresponding set of triangles in the mesh are then such that no two share a side, and thus the synchronization issues are circumvented.

The mesh is partitioned into a number of subdomains equal to the number of processors. Each processor "owns" the triangles in one subdomain. In addition, it has a copy of the unowned triangles that share a side with an owned triangle, known as a *ghost layer*. The ghost layer maintains information about changes in the mesh that have been made by neighboring processors. Their parallel algorithm is then

```
mark a set of triangles for refinement, based on local error estimates
repeat
    select an independent set, $I$, from marked triangles and incompatible triangles
    bisect marked triangles in $I$ by a longest edge
    bisect incompatible triangles in $I$ by a noncompatible edge
    exchange information about triangle refinement near the partition boundary with neighboring processors
until all marked triangles are refined and the mesh is compatible
```

Full Domain Partition

A different approach to parallel adaptive refinement was given by Mitchell [17] in 1998. The mesh is again partitioned into subdomains with one subdomain assigned to each processor, but here the ghost triangles extend over the whole domain. This is called the *full domain partition* (Figure 11). It adds the minimum number of triangles possible to cover the whole domain with a compatible mesh. The ghost triangles come from earlier refinement generations. The full domain partition was originally developed to reduce the frequency of communication between processors in a parallel multigrid algorithm, but it also fits well with adaptive refinement.

Mitchell proved that with a uniform mesh the number of ghost triangles on each processor is $O(\sqrt{M})$ where M is the number of owned triangles. This is of the same order as a single ghost layer, but with a larger constant. He conjectured that with adaptive meshes it is still $O(\sqrt{M})$ since there are $O(\sqrt{M})$ ghost triangles at the partition boundary, and the number of them decreases exponentially as you move away from the partition boundary. This was backed up with numerical evidence.

Consider the following sequential algorithm for adaptive refinement:

```
repeat
    mark a set of triangles for refinement based on local error estimates
    bisect marked triangles and those on the compatibility chains
until enough refinement (e.g. double the number of vertices)
```

With each processor having its mesh given by the full domain partition, the sequential algorithm only needs a slight modification for parallel execution. The changes are highlighted with italics.

```
repeat
    mark a set of owned triangles for refinement based on local error estimates
    bisect marked triangles and those on the compatibility chains
until enough refinement (e.g. reduce the maximum local error estimate by half)
reconcile differences between processors' meshes
```

There are three differences between the sequential and parallel algorithms. First, only owned triangles are marked for refinement. This is the only difference in the refinement loop. It is as if the processor is refining the mesh on the whole domain, but the local error estimate is zero on the ghost elements.

Second, the termination condition for the loop should be one that does not require communication between the processors. In the sequential algorithm we illustrate one that would require communication. A processor cannot know if the number of vertices in the global mesh has doubled without a global summation of the number of vertices owned by each processor. But in the parallel algorithm we illustrate one that does not require communication each time through the loop. There would be one communication step before the loop starts to determine the maximum local error estimate, but after that it is a local process to determine if all the local error estimates of owned triangles are less than half of the initial maximum local error estimate. Note that with this there is no communication at all in the main refinement loop.

Third, there is a communication step after the refinement loop to reconcile any differences between each processor's mesh. These differences arise when a compatibility chain extends beyond the owned partition into the ghost triangles. The owner of a refined ghost triangle must be made aware of that refinement and refine it if it has not already done so. A simple approach is to make a list of refined ghost triangles, exchange the lists between the processors, scan the received lists for owned elements and refine them if necessary. This only uses one communication step, but it is all-to-all. That is OK for a small number of processors, but quickly bogs down as the number of processors increases. A second approach uses only nearest neighbor communication, but may require more than one communication step. Given knowledge of the owner of each triangle in the first ghost layer, we have the following algorithm:

```
repeat
    make a list of refined elements in the first ghost layer for each neighbor processor
    exchange lists with neighbors
    refine triangles in the received lists, if not already refined, and those needed for compatibility
until the global mesh is compatible
```

We only need to send triangles in the first ghost layer because any others are in a compatibility chain starting with a refined triangle in the first ghost layer, and will be naturally refined by the neighbor. The loop is needed because it is possible for the compatibility chains within the loop to extend beyond partition boundaries.

There has been other research that is similar to or based on the full domain partition. Brandt and Diskin [18] published an algorithm in 1994 that has similarities to the full domain partition. This was not for adaptive refinement, but for parallel multigrid for finite differences on a uniform mesh. In 2000, Bank and Holst [19] presented "a new paradigm for parallel adaptive meshing algorithms" that was inspired by the full domain partition. It is very similar, but differs in some of the details. Currently Bank, Falgout, Jones, Manteuffel, McCormick and Ruge [20] are developing a new approach to algebraic multigrid that is "similar in spirit" to the full domain partition, but algebraic rather than geometric.

Tetrahedra

Zhang [21] presented a parallel algorithm for newest vertex bisection of tetrahedra in 2009. This is a parallel implementation of the refinement algorithm of Arnold et al. [15]. The domain is partitioned as usual, but with no ghost triangles. He recommends communicating the data as needed rather than using a ghost layer. This is a two phase refinement algorithm. In the first phase, elements in the processor's local mesh that are marked for refinement are bisected, as well as those in the compatibility chain within the local mesh. Faces on the partition boundary are treated like faces on the domain boundary. The second phase synchronizes between the processors for global compatibility. Neighboring processors exchange information about refined faces on the partition boundary, and each processor refines elements at the boundary as necessary. This process is repeated until the global mesh is compatible.

CONCLUSION

In 1985 Prof. Skeel raised the question of whether there was a way to perform adaptive refinement on a mesh of triangles without the red-green patching/unpatching process. This simple question has led to much research in newest vertex bisection adaptive mesh refinement and related fields by many people over the ensuing 30 years.

REFERENCES

1. R. E. Bank, and A. H. Sherman, *Computing* **26**, 91–105 (1981).
2. W. F. Mitchell, *Unified Multilevel Adaptive Finite Element Methods for Elliptic Problems*, Ph.D. thesis, University of Illinois at Urbana-Champaign, Urbana, IL (1988).
3. W. F. Mitchell, *J. Comp. Appl. Math.* **36**, 65–78 (1991).
4. G. E. Sewell, *Automatic Generation of Triangulations for Piecewise Polynomial Approximation*, Ph.D. thesis, Purdue University, West Lafayette, IN (1972).
5. J. Petersen, *Acta Mathematica* **15**, 193–220 (1891).
6. M. R. Garey, and D. S. Johnson, *Computers and Intractability: a Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, 1979.
7. T. Biedl, P. Bose, E. D. Demaine, and A. Lubiw, *J. Alg.* **38**, 110–134 (2001).
8. M.-C. Rivara, *SIAM J. Numer. Anal.* **21**, 604–613 (1984).
9. M.-C. Rivara, and C. Levin, *Comm. Appl. Numer. Methods* **8**, 281–290 (1992).
10. E. Bänsch, *J. Comput. Appl. Math* **36**, 3–28 (1991).
11. I. Kossaczky, *J. Comput. Appl. Math* **55**, 275–288 (1994).
12. A. Liu, and B. Joe, *SIAM J. Sci. Comput.* **16**, 1269–1291 (1995).
13. J. M. Maubach, *SIAM J. Sci. Comput.* **16**, 210–227 (1995).
14. C. T. Traxler, *Computing* **59**, 115–137 (1997).
15. D. N. Arnold, A. Mukherjee, and L. Pouly, *SIAM J. Sci. Comput.* **22**, 431–448 (2000).
16. M. T. Jones, and P. E. Plassmann, *SIAM J. Sci. Comput.* **18**, 686–708 (1997).
17. W. F. Mitchell, "The Full Domain Partition Approach to Parallel Adaptive Refinement," in *Grid Generation and Adaptive Algorithms, IMA Volumes in Mathematics and its Applications*, Springer-Verlag, 1998, vol. 113, pp. 151–162.
18. A. Brandt, and B. Diskin, *Contemporary Mathematics* **157**, 135–155 (1994).
19. R. E. Bank, and M. Holst, *SIAM J. Sci. Comput.* **22**, 1411–1443 (2000).
20. R. Bank, R. Falgout, T. Jones, T. Manteuffel, S. McCormick, and J. Ruge, *SIAM J. Sci. Comput.* **37**, 113–136 (2015).
21. L. bo Zhang, *Numer. Math. Theory, Methods, Appl.* **2**, 65–89 (2009).